VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Možnosti technologie Adobe AIR

Potential of Adobe Integrated Runtime (AIR)

2010 Radomír Benk

Prohlašuji, že jsem byl seznámen s tím, že na moji bakalářskou/diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.

Beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě bakalářskou/diplomovou práci užít (§35 ods. 3).

Souhlasím s tím, že jeden výtisk bakalářské/diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a údaje o bakalářské /diplomové práci budou zveřejněny v informačním systému VŠB-TUO.

Beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě, 7.5.2010 .....

Radomír Benk

Adresa trvalého pobytu studenta:
Slunná 27, 78901 Zábřeh, Česká republika

Abstrakt

Tato práce se zabývá zobrazením potenciálu technologie Adobe AIR a jejím porovnáním s některými dalšími technologiemi týkajícími se RIA problematiky.

V úvodní části práce jsou uvedeny a definovány některé základní pojmy týkající se RIA aplikací a celkově webových aplikací fungujících hlavně na klientské straně. V další části práce jsou zobrazeny některé technologie určené pro vývoj RIA aplikací, a jejich následné zhodnocení a porovnání výhod a nevýhod. Dále je zde zobrazen vývoj AIR aplikace, na které jsou demonstrovány možnosti technologie Adobe AIR.

Klíčová slova: Adobe AIR, AJAX, Silverlight, JavaFX, Flash, Flex, RIA, webová aplikace

Abstract

This work consider about potential of Adobe AIR technology and it’s comparing with some other RIA issues technologies.

In the first part of the work there are named and defined some basic terms relating RIA appliacations and overall web applications working mainly on client side. In the next part of the work there are showed some technologies specified for developing RIA applications, and theirs evaluation and comparion of advantages and disadvantages. Next there is showed development of AIR application, which demostrates potential of Adobe AIR.

Keywords: Adobe AIR, AJAX, Silverlight, JavaFX, Flash, Flex, RIA, web appliaction

Čestné prohlášení

Prohlašuji čestně, že jsem tuto práci zpracoval samostatně s využitím literatury zde uvedené.

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Mgr. Zdeňku Horákovi za cenné rady, připomínky, vedení a zároveň volnou ruku při vytváření této práce.

Obsah:

1. Úvod

Webové technologie se čím dál více rozrůstají a zvětšují. Již dávno neplatí, že webová aplikace musí být program zpracovávaný někde na serveru. Z důvodů většího pohodlí uživatele, vyšší míry interakce a menších nároků na server se webové aplikace přesouvají ke zpracování ze serveru na klienta. Hlavně díky zpříjemnění práce klienta mají tyto aplikace snahu podobat se aplikacím desktopovým, jejichž ovládání už je uživateli známé. Tyto webové aplikace se ovšem vyvíjejí do podoby desktopových aplikací stále víc a pomalu se začínají osamostatňovat a odpoutávat od na začátku vytyčeného webového prohlížeče.

Vzniká tak zcela nový druh aplikací, které jsou tvořeny webovými technologiemi, ale zároveň pracují na klientské straně a nemusí nutně vyžadovat připojení k serveru, tzn., že mohou pracovat „offline“.

Cílem této práce je popsání některých takovýchto technologií, zobrazení jejich výhod a nevýhod, možností, a porovnání s technologií Adobe AIR (Adobe Integrated Runtime).

Dále je představena tato technologie na demonstrační aplikaci, jsou zobrazeny prvky shodné s desktopovou aplikací, popsání výhod a nevýhod této technologie a celkové zhodnocení.

2 Základní pojmy

Cílem této kapitoly je vysvětlení několika pojmů, se kterými se bude dále pracovat. Jedná se pouze o upřesnění termínů jinak často používaných.

2.1 Statický / dynamický web

Statickou stránkou je rozuměn dokument, se kterým nemůžete nijak pracovat, neobsahuje žádné interaktivní prvky. Jedná se pouze o holý dokument, který nevyužívá žádné prvky, které by mohl uživatel nějak upravovat, přidávat, či ubírat. Svým způsobem se jedná o „chudou stránku“. Jedinou funkcí statické stránky je tedy zobrazení předem vytvořeného přeneseného dokumentu ze serveru na klienta.

Dynamický web naproti tomu obsahuje nějaké vylepšení, nebo nějaký složitější prvek, který umožní uživateli provádět nějaké akce, které se netýkají pouze procházení dokumentu. Dynamický web umožňuje interakci uživatele, rozpohybování dokumentu. Pro tyto operace je využit jazyk HTML jakožto základní stavební kámen, který je doplněn o scriptovací jazyk, nebo nějakou jinou technologii určenou k rozpohybování stránek a jejich oživení.

2.2 Tenký / Tlustý klient

- Tlustý klient (Fat client): [1]
  Aplikace se realizuje na straně klienta, využití připojení k serveru se předpokládá pouze pro přístup k datům. Většina výpočetních operací se tedy provádí na klientovi, což má za následek menší nároky na server a větší nezávislost klienta na serveru.

- Tenký klient (Thin client): [2]
  Tento typ připojení je plně závislý na připojení k serveru. Všechny výpočty a operace se provádějí na serveru a výsledek je přenesen na klienta, kde se zobrazuje. Tento způsob připojení je závislý na výkonu serveru, jeho výhodou je jednodušší zabezpečení a vylepšená bezpečnost dat (data jsou uchovávána na serveru, nikoli na klientovi).

2.3 Webová aplikace

Na Wikipedii lze vyhledat poměrně jednoznačnou definici pro tento termín:[3] „Webová aplikace je aplikace poskytována uživatelům z webového serveru přes síť Interner (nebo vnitřní síť ethernet).“ Cílem takovéto aplikace je nabídnout klientům služby bez předchozího instalování jakéhokoliv složitého softwaru a využití softwaru již používaného – tedy třeba internetový prohlížeč. Důležitá je právě architektura klient/server.

Takovéto aplikace pracují hlavě na straně serveru a klientskému prohlížeči se nechávají pouze některé jednodušší výpočetní operace, případně úprava grafického vzhledu. Klient má možnost se serverem nějakým způsobem spolupracovat. Pro komunikaci mezi klientem a serverem se využívá protokol HTTP a koncovým rozhraním pro uživatele je webový prohlížeč. Struktura webové aplikace odpovídá struktuře Tenkého klienta.

Pro příklad si můžeme představit obsluhu e-shopu. Uživatel ovládá pouze některé prvky, může zadat kolik, a co si chce objednat, ovšem všechny transakce a objednávací operace probíhají výhradně na serveru.

2.4 HTTP protokol (HyperText Transfer Protocol)

Přesnou definici HTTP protokolu lze nalézt v RFC 2616 [4]. Protokol definuje komunikaci klienta se serverem, nejčastěji přes port 80. Protokol je textový a bezestavový – protokol neumí uchovávat stav komunikace, dotazy spolu nemají souvislost.

Po připojení k serveru dojde k otevření socketu a je odeslán HTTP požadavek. Hlavička požadavku může vypadat takto:

```
GET / HTTP/1.0
Host: www.google.com
User-Agent: Mozilla/5.0
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: cs-CZ,cs;q=0.9,en-US;q=0.8,en;q=0.7,defaultQLS
Accept-Encoding: gzip,deflace
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.7
```

Server následně požadavek zpracuje a odpoví:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Content-Encoding: zip
Server: gws
Content-Length: 68
```

Po přijetí klient zpracuje přijatá data a zobrazí nejčastěji ve webovém prohlížeči, odkud byl požadavek odeslán.

2.5 FTP Protokol (File Transfer Protocol)

Přesnou definici FTP protokolu lze nalézt v RFC-959 [5].

FTP je protokol fungující na principu klient-server, určen pro přenos souborů mezi počítači. Funguje přes TCP (Transmission Control Protocol), a to na portech 20 a 21. Port 21 je řídící – přenášejí se zde příkazy určené k obsluze směrem na server a následně odpovědi ze serveru na klienta. Port 20 je datový – určený k přenosu pouze dat z, nebo na server.

Uživatel tedy zadává příkazy přes port 21. Ve chvíli, kdy je navázáno spojení, a jsou vybrány soubory určené k přenosu, se otevře port 20, po kterém jsou data přenášena.

FTP běží buďto v aktivním, nebo pasivním připojení:

- V aktivním režimu vytváří spojení pro přenos dat server a klient „naslouchá“.
- V pasivním režimu server odešle klientovi svou IP adresu a port, na kterém naslouchá, a klient vytváří datové spojení.

Pasivní režim je složitější na realizaci, ovšem díky tomu, že vytváří připojení klient, nenastává problém, pokud je klient v privátní síti, a navíc je pasivní režim bezpečnější.

Následuje příklad ukazující připojení k serveru a stažení souboru. Uživatel „Klient“ je požádán o heslo, to je ověřeno a následně je zavolán příkaz RETR, který otevře datové spojení. Po celou dobu

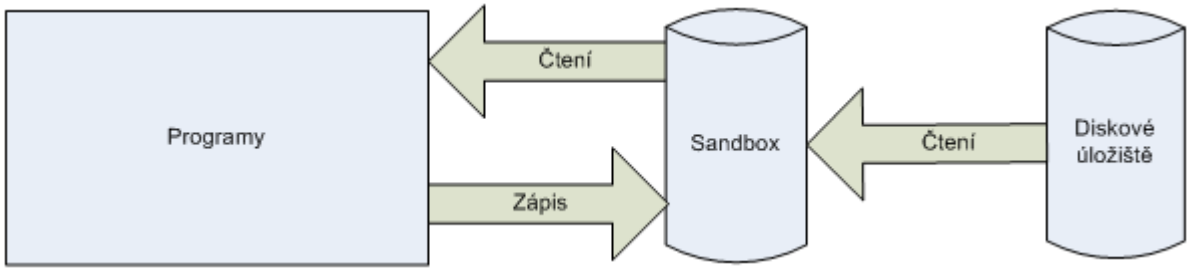
je udržován pasivní režim, který si vždy vyžádá IP adresu serveru a číslo datového portu. Modře jsou znázorněny zprávy směřující na server, zeleně poté zprávy určené klientovi.

```
(not logged in) (127.0.0.1)> Connected, sending welcome message...
(not logged in) (127.0.0.1)> 220-FileZilla Server version 0.9.34 beta
(not logged in) (127.0.0.1)> 220-written by Tim Kosse (Tim.Kosse@gmx.de)
(not logged in) (127.0.0.1)> 220 Please visit
  http://sourceforge.net/projects/filezilla/
(not logged in) (127.0.0.1)> USER Klient
(not logged in) (127.0.0.1)> 331 Password required for Klient
(not logged in) (127.0.0.1)> PASS ****
Klient (127.0.0.1)> 230 Logged on
Klient (127.0.0.1)> PASV
Klient (127.0.0.1)> 227 Entering Passive Mode (127,0,0,1,126,76)
Klient (127.0.0.1)> RETR /HelloWorld.html
Klient (127.0.0.1)> 150 Connection accepted
Klient (127.0.0.1)> 226 Transfer OK
Klient (127.0.0.1)> PASV
Klient (127.0.0.1)> 227 Entering Passive Mode (127,0,0,1,126,78)
Klient (127.0.0.1)> QUIT
Klient (127.0.0.1)> 221 Goodbye
Klient (127.0.0.1)> disconnected.
```

2.6 Sandbox

Sandbox (pískoviště) je speciální chráněný prostor, ve kterém může program provádět jakoukoliv činnost, aniž by narušil bezpečnost celého systému. Data se nezapisují přímo na disk, ukládají se pouze do vyrovnávací paměti. K důležitým souborům a nastavením operačního systému se však aplikace běžící v sandboxu nedostane. Je to velmi efektivní způsob, jak zamezit přístupu k nechtěným datům. Zároveň to však může omezit přístup k lokálním zdrojům.

Lépe je to ukázáno na obrázku 2.6.1, kde je znázorněn sandbox a jeho vztahy s programy a úložištěm dat.



Obrázek 2.6.1 Přístup sandboxu k programům.

2.7 RIA (Rich Internet Application)

Rich Internet Application (v překladu Bohatá Internetová Aplikace) je termín označující webovou aplikaci, která má nějaké charakteristické rysy desktopové aplikace.

Na obrázku 2.7.1 vidíme oblast, kterou pokrývá RIA.

Čím dále častěji je požadována dostatečná grafická úroveň, nejlépe aby nevybočovala z již zavedených trendů. Pro uživatele je daleko komfortnější, když narazí na něco známého, na co už je zvyklý a alespoň intuitivně zvládne ovládání, než kdyby se musel učit novým přístupům a způsobům ovládání. Proto jsou webové aplikace co nejvíce směřovány, tak aby nesly alespoň některé znaky

společné pro desktopové aplikace, ať už se jedná o klávesové zkratky, tvary oken, druhy, umístění menu, nebo nápovědu.

Obrázek 2.7.1 Zobrazení vztahů RIA k ostatním technologiím

Internetové aplikace jsou v tomto směru poměrně hodně limitovány jednak protokolem HTTP. – např. komunikace klient / server, při každé změně vyvolané klientem. Druhou slabinou je grafické rozhraní – HTML, případně CSS nenabízí dostatečné možnosti přiblížení se grafickým podobám desktopových aplikací, a hlavně dostatečný komfort pro uživatele. Mnoho RIA aplikací proto využívá svůj vlastní runtime software (tedy běhové prostředí), který je potřeba doinstalovat na klientském stroji pro použití dané aplikace.

Další změnou oproti webové aplikaci je práce klienta / serveru. Díky tomu, že klient využívá další software pro běh aplikace, se může většina práce přesunout na stranu klienta a server již není tak moc zatěžován spoluprácí s klientem. Z toho také plyne další výhoda – a to samostatnost klienta.

Klient může teoreticky využívat danou aplikaci, i pokud není připojen k internetu. Samozřejmě pokud běh aplikace vyžaduje data z databáze serveru, nebo nějakou spolupráci se serverem, tak to možné není. Ale pokud aplikace v danou chvíli nic takového nevyžaduje, nic nebrání klientovi v prácí s danou aplikací. Což představuje velkou výhodu právě pro mobilní zařízení, která nemusejí mít nutně stálé připojení do sítě internet.

RIA aplikace tedy fungují jako Fat-client. V souvislosti s RIA je ovšem tento typ klienta označován jako Rich-client.

2.8 JavaScript (JS)

Detailní popis funkčnosti lze nalézt na oficiálních stránkách JavaScriptu [6].

JavaScript je multiplatformní skriptovací programovací jazyk určený pro řešení dynamiky WWW stránek na klientské straně. Používá se obvykle na vkládání různých interaktivních prvků (tlačítka, textová pole). Syntaxí je podobný jazykům C/C++, Java. Důležitou vlastností je beztypovost datových typů.

Je závislý na interpretačním prostředí (v každém prohlížeči může fungovat trochu odlišně).

```
<script type="text/javascript">
      for (var = 0; i < 5; i++)
      {
          alert("Hello for the"+ (i+1)+“. time“);
      (
</script>
```

JavaScript také může ovládat a pohybovat s DOM objekty umístěnými na stránce, což vytváří právě onen interaktivní pocit pohybu.

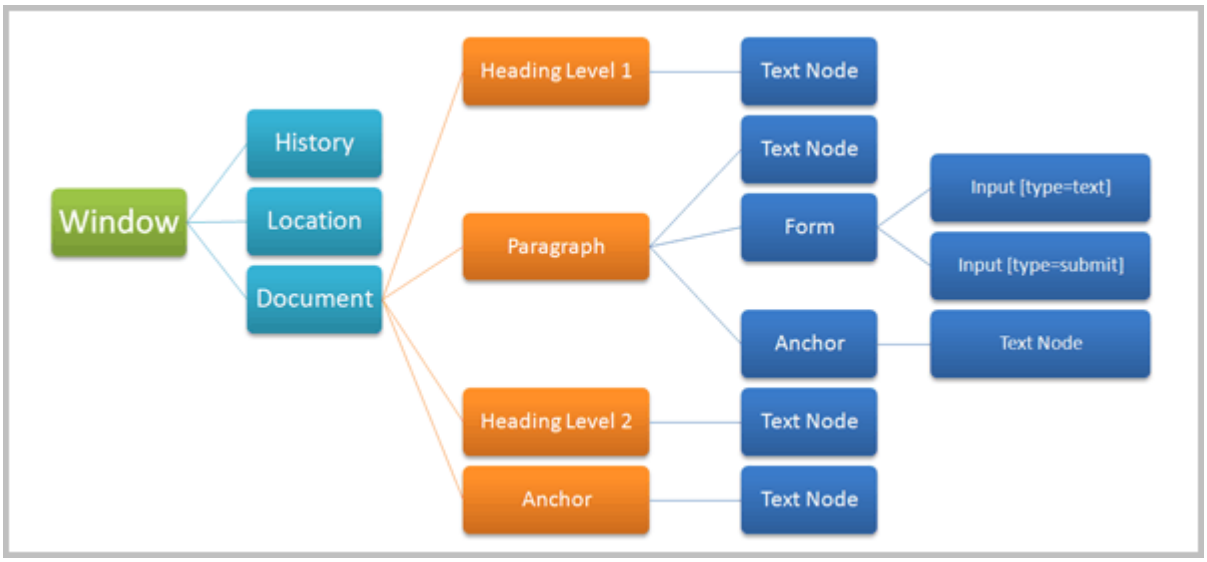
JavaScriptové frameworky jsou JavaScriptové knihovny určené ke zjednodušení a rozšíření standardního JavaScriptu. Jsou to skripty napsané v JS, které rozšiřují stávající objekty. Rozlišujeme dvě skupiny frameworků:

- JavaScriptové knihovny – rozšiřují funkcionalitu (Prototype, jQuery).
- RIA frameworky – komplexní řešení RIA (YUI, Dojo, ExtJS).

2.9 DOM (Document Object Model)

Na stránkách W3.org lze nalézt kompletní dokumentaci k DOM [7].

DOM je objektově orientovaná reprezentace XML nebo HTML dokumentu. Umožňuje přístup k jednotlivým částem obsahu, nebo stylu dokumentu. Z jednotlivých elementů dokumentu je vytvořena stromová struktura. Díky tomu se udržuje příslušná hierarchie. DOM byl zaveden z důvodu standardizace HTML elementů.



Obrázek 2.9.1 Zjednodušená struktura DOM [8]

Na obrázku 2.9.1 je ukázána zjednodušená hierarchie a propojení objektů v dokumentu.

2.10 JSON (JavaScript Object Notation)

JSON[9] je datový formát nezávislý na platformě určený pro přenos dat tvořící strukturu. JSON je zcela obecný a může být použit kterýmkoliv jazykem. Nevýhodou je nemožnost definovat znakovou sadu. Alternativou je XML, které může nějak definovat obsah dat, obsahuje ale v průměru 40% dat ve značkách, proto je JSON považován za odlehčenou alternativu XML.

```
var myJSON =
  {
    'color' : 'blue',
    'animal' : 'dog'
  }
};
Alert(myJSON.color);
```

Jak vidíme na příkladu, k proměnné „color“ definované v objektu myJSON se přistupuje stejně jako k vlastnosti objektu. Výstupem tohoto kódu bude string „blue“.

2.11 ExtJS

Kompletní strukturu a dokumentaci lze nalézt na oficiálních stránkách ExtJS [10].

ExtJS je Javascriptový framework určený k vytváření vysoce interaktivního vzhledu. Obsahuje předdefinované komponenty, které se nastavují a upravují podle parametrů. Je tedy objektově orientovaný. Na příkladu kódu můžeme vidět vytvoření jednoduchého formuláře s několika prvky. Lze vidět, že u prvků můžeme nastavit poměrně hodně parametrů.

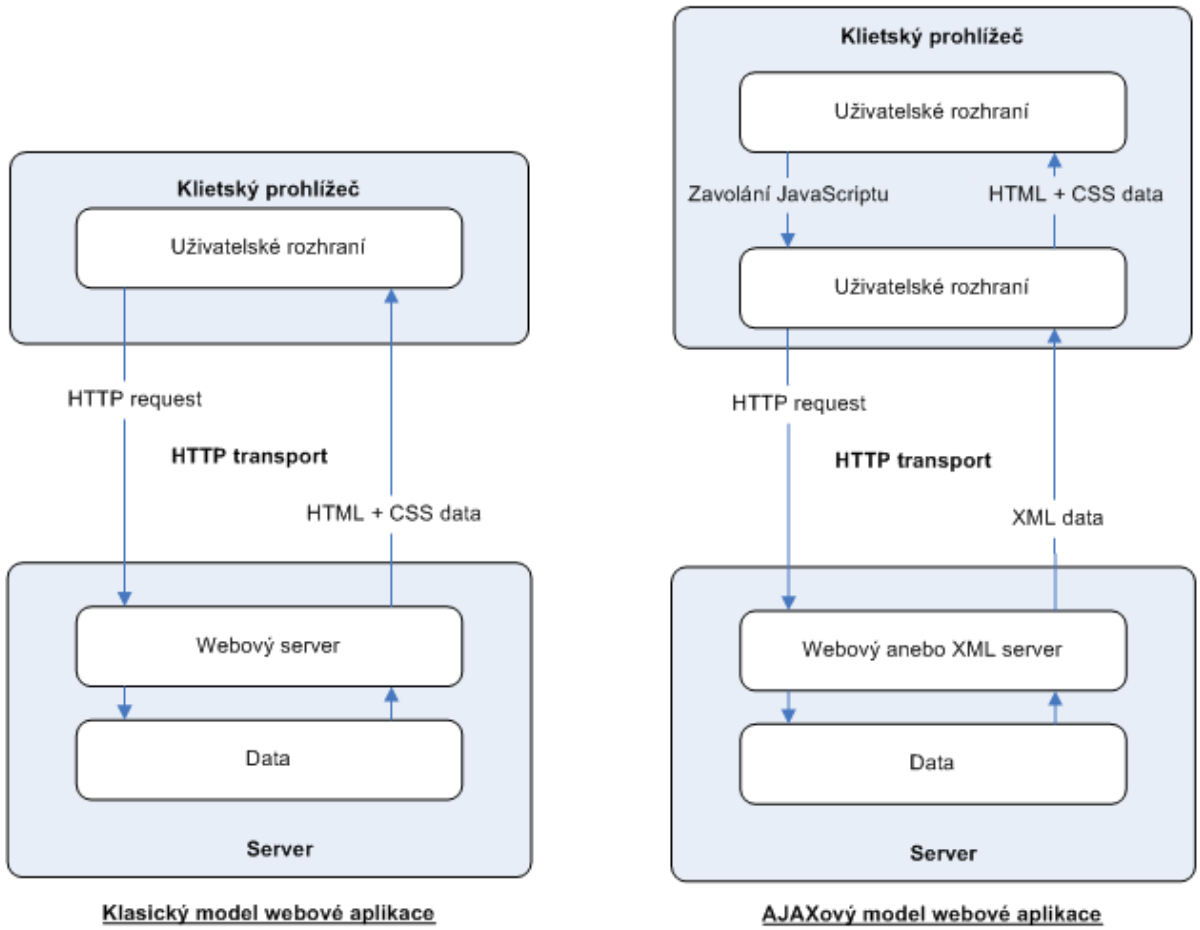
```
var simple = new Ext.FormPanel({
        labelWidth: 75,
        frame:true,
        title: 'Simple Form',
        bodyStyle:'padding:5px 5px 0',
        width: 350,
        defaults: {width: 230},
        defaultType: 'textfield',
        items: [{
                fieldLabel: 'Name',
                name: 'first',
                allowBlank:false
            },{
                fieldLabel: 'Company',
                name: 'company'
            }, {
                fieldLabel: 'Email',
                name: 'email',
                vtype:'email'
            }, new Ext.form.TimeField({
                fieldLabel: 'Time',
                name: 'time',
                minValue: '8:00am',
                maxValue: '6:00pm'
            })
        ],

        buttons: [{
            text: 'Save'
        },{
            text: 'Cancel'
        }]
    });
```

3 RIA technologie

V této kapitole se budeme snažit zhodnotit některé technologie používané pro vývoj RIA aplikací, jejich klady, zápory a možnosti.

3.1 AJAX (Asynchronous JavaScript And XML)

AJAX (Asynchronous JavaScript and XML) [11] je skupina technologií pracující na klientské straně webové aplikace. Umožňuje získávání dat ze serveru asynchronně na pozadí načtené stránky bez nutnosti obnovy stránky, což má za následek větší dynamičnost, menší nároky na přenos dat (není nutnost načítat celou stránku, ale pouze určitou vyžádanou část).

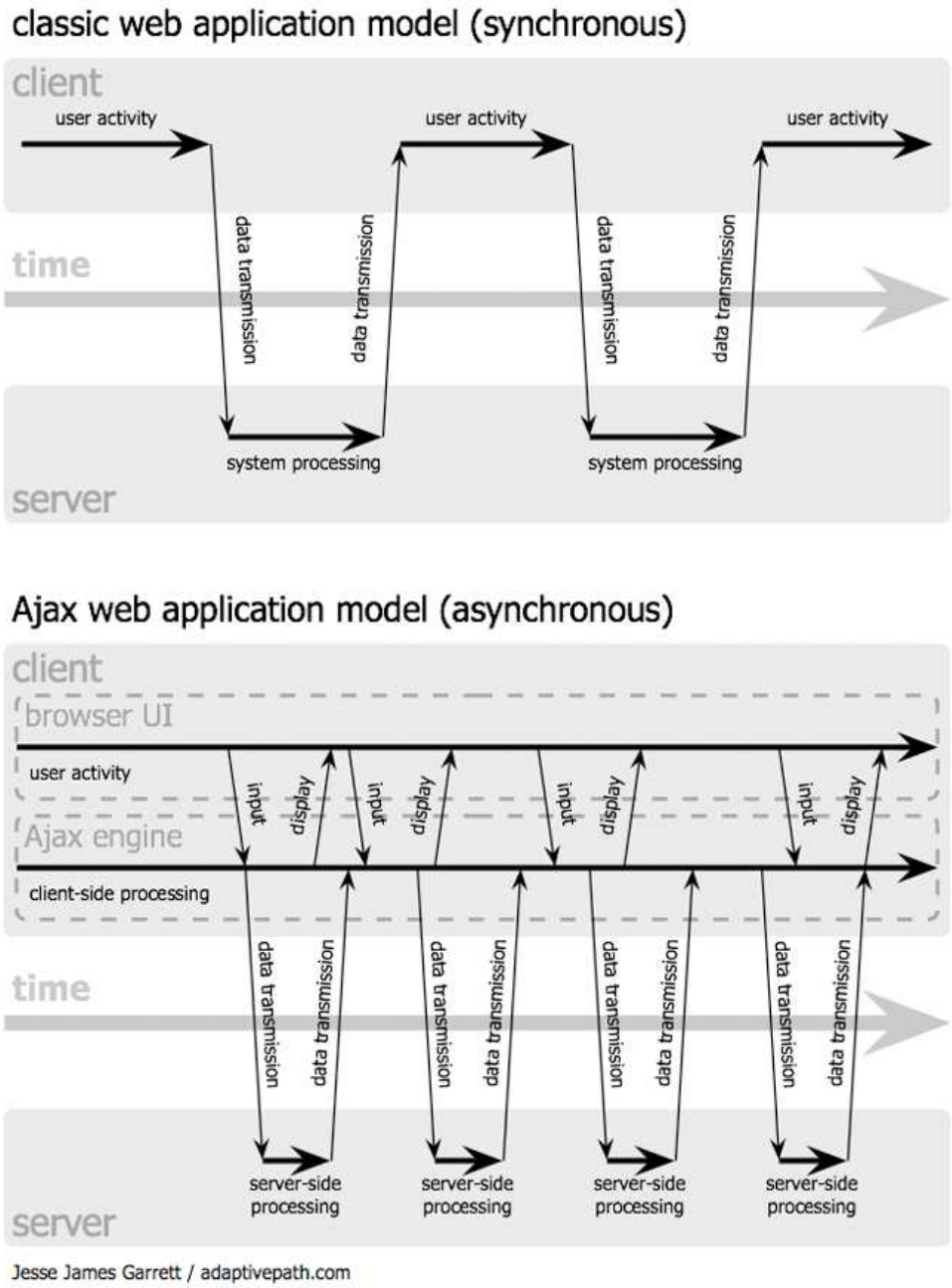


Obrázek 3.1.1 Znázornění AJAX modelu proti klasickému zobrazení

Na obrázku 2.1.1 je znázorněna komunikace klienta se serverem. Bez použití AJAXu (classic web application model) klientské uživatelské rozhraní komunikuje přímo se serverem – všechny operace jsou odesílány na server, zde jsou spracovány a vráceny v podobě nové html stránky.

Oproti tomu při použití AJAXu (Ajax web application model) klientské uživatelské rozhraní komunikuje s AJAX vrstvou přes JavaScript, AJAX vrstva poté podle potřeby a zadaných instrukcí od uživatele získá ze serveru pouze specifická konkrétní data (Typicky jsou data ve formátu XML, ale

může být získán i jiný typ formátu, např. HTML, čistý text, JSON, EBML). Daná data jsou AJAXem zpracována, a poté vložena do webového prohlížeče klienta.



Obrázek 3.1.2 Znázornění komunikace přenosu dat mezi serverem a klientem [12]

Na obrázku 3.1.2 je vidět, že komunikace mezi klientem a serverem bez použití AJAXu (classic web application model) je plně synchronní, tedy uživatel musí čekat na odpověď serveru, a dokud mu odpověď nepřijde, nemůže provádět žádnou operaci. Zatímco při použití AJAXu může uživatel pracovat s prohlíženou stránkou, zatímco data jsou získávána ze serveru a zpracovávána na pozadí.

Zde je ukázka zpracování AJAXu v JavaScriptu. Při zavolání funkce „sayHello()“ se odešle požadavek na server a nastaví se event listener čekající na odpověď („handleSayHello()“). Jakmile je získána odpověď ze serveru, je vypsána zpráva s textem odpovědi.

```
var receiveReq = getXmlHttpRequestObject();

function sayHello()
{
  receiveReq.open("GET", 'SayHello.html', true);
  receiveReq.onreadystatechange = handleSayHello;
  receiveReq.send(null);
}

function handleSayHello()
{
  Alert(receiveReq.responseText);
}
```

3.1.1 AJAX v sobě zahrnuje následující technologie:

- HTML nebo XHTML a CSS pro standardní základní zobrazení
- DOM (Document Object Model) pro dynamické zobrazování a interakci
- XML a XSLT pro manipulaci s daty
- XMLHttpRequest pro asynchronní získávání dat
- JavaScript pro provázání všech technologií dohromady

3.1.2 Výhody:

- Odstranění nutnosti opětovného načítání stránky – AJAX umožňuje zpracování pouze části stránky, která je díky manipulaci s DOM aktualizována přímo na klientské straně.
- Z důvodu získávání pouze konkrétních vyžadovaných dat se snižuje zátěž a nároky na server a množství přenesených dat. Komunikace serveru s klientem je díky tomu teoreticky zrychlena.
- Díky asynchronní komunikaci vzniká pocit větší plynulosti a tedy i větší uživatelský komfort.

3.1.3 Nevýhody:

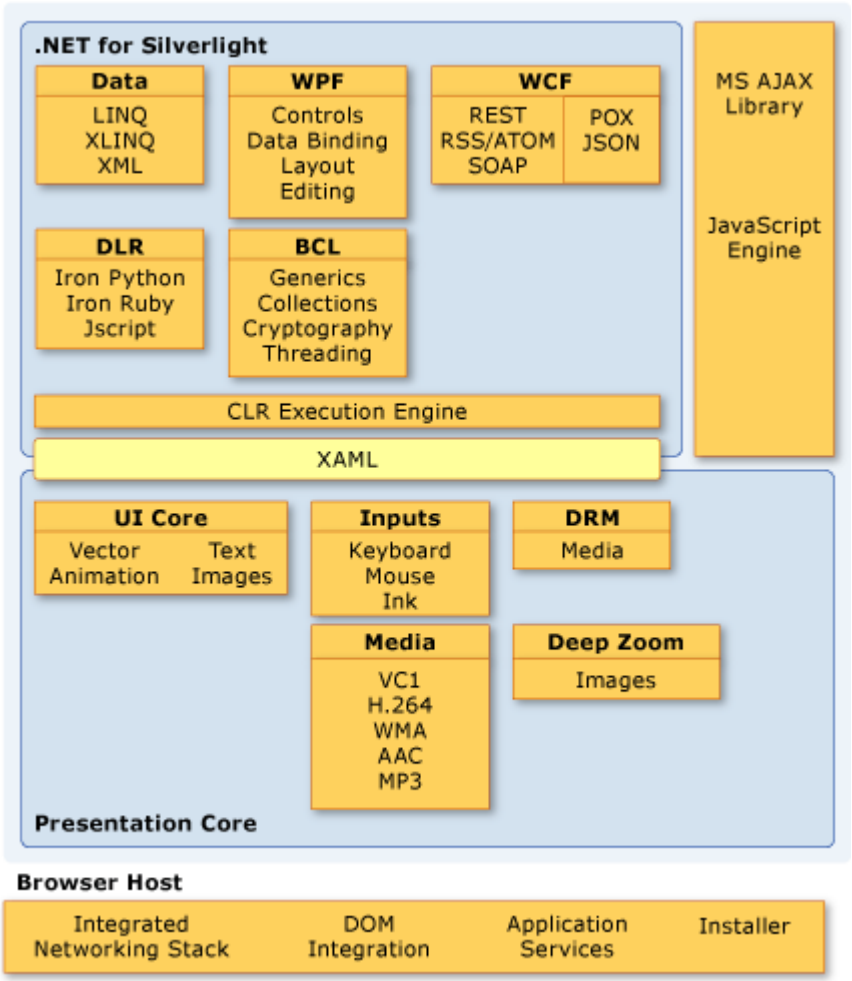
- Stránka se chová jako plnohodnotná webová aplikace – všechny operace jsou prováděny na AJAX vrstvě, tedy na klientovi, to má za následek následující nevýhody:
    - Tlačítka „Zpět“ a „Další“ v prohlížeči nefungují zrovna nejlépe. Protože jsou operace prováděny uvnitř aktuální stránky, neexistuje žádná předchozí nebo další stránka. Řešení se dá doprogramovat přímo v kódu AJAXu přidáním alespoň částečné historie – to znamená další programování navíc.
    - Nelze předávat „naklikaná“ data na stránce přes URL, protože samotné URL daná data neobsahuje – jsou obsažena pouze na klientské vrstvě AJAXu. Toto se dá taktéž alespoň z části obejít využitím prostoru za názvem adresy (za znakem #), ale samozřejmě s tím stoupá náročnost aplikace.
    - Nutnost mít zapnutý JavaScript v prohlížeči.
    - U starších prohlížečů může dojít vlivem JavaScriptu k větší zátěži počítače.

3.2 Microsoft Silverlight

Microsoft Silverlight[13] je velmi mocná technologie snažící se proniknout na všechny platformy, fungující v klientském prohlížeči (a od verze 3.0 i mimo prohlížeč) a umožňující spojení WPF (Windows Presentation Foundation) s .NET Frameworkem. Díky tomu vzniká velmi silný nástroj umožňující vytvářet plnohodnotné klientské aplikace. Silverlight vznikl jako reakce na všudypřítomný Flash od společnosti Adobe, který fungoval velmi dlouhou dobu jako monopol.

Základní rysy jsou:

- Podpora rastrové i vektorové grafiky.
- Podpora multimedií a animací.
- Podpora GUI prvků.
- Podpora síťových technologií.
- Využívání sandboxu.
- Lokální úložiště dat a oddělení aplikační logiky od dat.



Obrázek 3.2.1 Struktura WPF a Silverlight [14]

3.2.1 WPF (Windows Presentation Foundation)

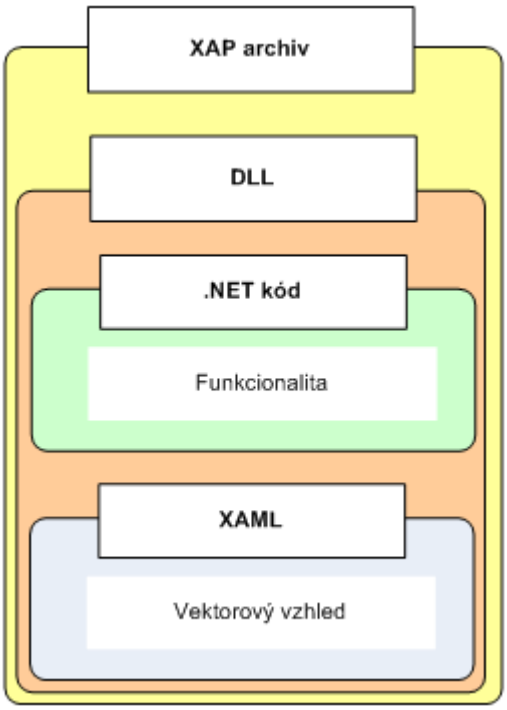
WPF (Windows Presentation Foundation) je grafický subsystém, který je součástí .NET Frameworku 3.0 pro vytváření uživatelských rozhraní. Hlavním důležitým rysem je poskytnutí programového modelu, který bude sjednocovat grafické prvky a oddělovat grafickou část a programovou logiku. K tomuto využívá značkovací jazyk XAML. Obrázek 3.2.1 znázorňuje propojení a sjednocení technologií uvnitř WPF.

3.2.2 XAML (eXtensible Application Markup Language)

XAML (eXtensible Application Markup Language) je deklarativní značkovací jazyk, který se používá hlavně k definování a vytváření grafických uživatelských rozhraní. Jednotlivé prvky uživatelského rozhraní se zde nadefinují a pojmenují a podle toho k nim přistupuje programová logika. Všechny prvky použité v XAML mohou být vytvořeny taktéž v jazyce .NET. Výhodou je ovšem to, že XAML vycházející z XML, je pro designéry daleko přijatelnější, než kód psaný v jazyce .NET. Následující příklad zobrazuje vypsání textu „Hello world“ v XAML.

```
<?xml version=”1.0″?>
<Canvas xmlns=”http://schemas.microsoft.com/client/2007″>
  <TextBlock Text=”Hello World!” />
</Canvas>
```

3.2.3 Struktura Silverlight



Obrázek 3.2.3.1 Spojení jednotlivých částí Silverlight

Na obrázku 3.2.3.1 je znázorněna struktura a propojení uvnitř aplikace Silverlight. Jasně zde můžeme vidět oddělení programové logiky od uživatelského rozhraní. Designér vytvoří uživatelské rozhraní nezávisle na obsahu kódu v .NET. Pomocí XAML jsou nadefinovány názvy prvků, ke kterým poté .NET kód přistupuje a operuje s nimi. Jedná se tedy o velmi jednoduchou cestu k vytvoření plnohodnotné aplikace pro znalce .NET.

3.2.4 Moonlight

Technologie Silverlight je závislá na platformě. Je určena pouze pro operační systémy Windows. Z důvodů velké kritiky, a hlavně pro ovládnutí dalších platforem, zadal Microsoft vývoj open-source implementace Silverlightu primárně pro Unixové platformy. Mono Project sponzorován Novellem začal s vývojem tohoto open-source – Moonlight [15]. Microsoft k tomuto poskytuje značnou podporu – kódy kodeků, detailní specifikaci k Silverlightu. Ve výsledku by měl být Moonlight schopen plně zastoupit Silverlight ve webových prohlížečích na jiných platformách.

Povolení vytvoření Moonlight technologie se ovšem může zdát jako dobrý tah pro Microsoft. Tím, že dá některé své technologie „do prostoru“, nastaví standard. Díky tomu bude moci ovládnout trh, protože bude vždy převyšovat tento standard a vždy bude moci nabídnout něco víc.

3.2.5 Výhody

- Podpora přehrávání videa, široká škála typů videa a například i HD, HD DVD, nebo Blu-ray.
- Jeden programovací jazyk pro vývoj celé aplikace (prohlížeč, server, databáze - .NET).
- Oddělení grafické části a programové logiky.
- Microsoft poskytuje velmi účinné, výkonné a efektivní nástroje pro tvorbu aplikace (Expression Blender + Visual Studio).

3.2.6 Nevýhody

- Platformou závislé. Toto je zčásti řešeno open-source technologií Moonlight.

3.3 JAVA

Kompletní přehled o technologii Java lze nalézt na oficiálních stránkách [16].

3.3.1 Java Applet

Java Applet [17] je program napsaný programovacím jazykem Java. Takto vytvořený a zkompilovaný kód v jazyce Java může být následně vložen do HTML stránky pomocí klíčového tagu APPLET.

Např:

```
<APPLET code="HelloWorld.class" WIDTH="200" HEIGHT="40">
```

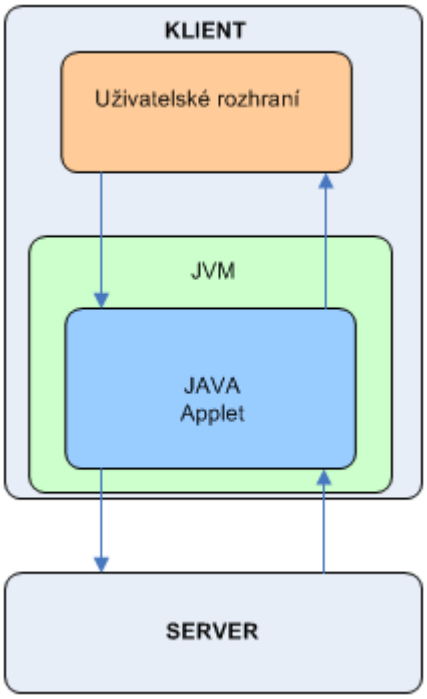
Samozřejmě zkompilovaných souborů může být více, a to může mít za následek velké zpomalení při stahování appletu. Proto jsou všechny třídy uloženy do archivu „jar“.

Např:

```
<APPLET code="HelloWorld" WIDTH="200" HEIGHT="40" ARCHIVE="HelloWorld.jar">
```

Uživatel si jej stáhne při načítání stránky, a jestliže má uživatel nainstalováno běhové prostředí – Java Virtual Machine (JVM), tak se applet spustí přímo v prohlížeči na dané HTML stránce. Zdrojový kód appletu v jazyce Java může vypadat takto:

```
import java.applet.Applet;
import java.awt.*;

public class HelloWorld extends Applet {
    public void init() { }

    public void stop() { }

    public void paint(Graphics g) {
        g.drawString("Hello, world!", 20,10);
  }
}
```



Obrázek 3.3.1.1 Zapojení appletu do struktury klient-server

Na obrázku 3.3.1.1 vidíme spojení klientského rozhraní se serverem. Hlavní podstatnou částí pro fungování appletu je přítomnost JVM na klientovi.

I když applet není nejpoužívanější technologií hlavně z důvodu rychlosti, jeho hlavní rysy se shodují s hlavními rysy RIA:

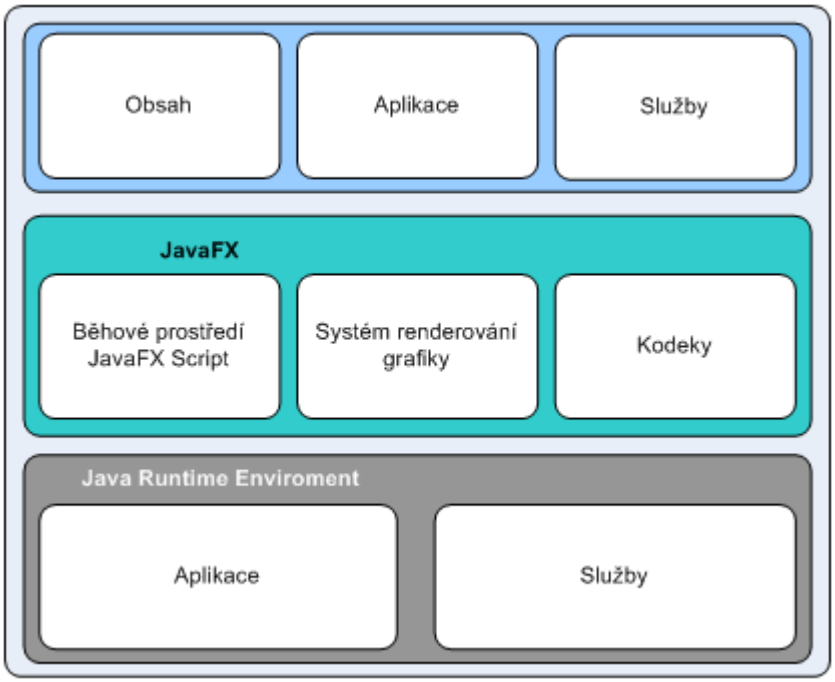
- Pro svůj běh využívá sandbox.
- Většina operací je prováděna na klientovi.

- Po stažení na klienta může fungovat i v offline režimu.
- Java applet je platformově nezávislý. Potřebuje pouze přítomnost JVM.

3.3.2 JavaFX

JavaFX [18] je klientská platforma určená k vývoji RIA. Zaměřuje se hlavně na grafické prvky a cílem je možnost soustředění se na grafickou úpravu namísto kódování. JavaFX byla vytvořena jako reakce na rychle se rozšiřující webové Adobe technologie a MS Silverlight.

- Je spouštěna formou pluginu v prohlížeči, ale je zde možnost spuštění i v offline módů přímo z desktopu.
- Multiplatformní technologie.
- Přenositelnost kódu mezi platformami s velkým využitím i na mobilních zařízeních (telefony, PDA). Obzvláště na Androidu, ale pracuje i na Windows Mobile.
- Aplikace je psaná v jazyce JavaFX Script, ovšem má možnost používat napsané třídy z jazyka Java.
- Podpora multimediálních prvků.
- Ochrana zdrojového kódu – tak jako u appletu se uživatel dostane pouze ke zkompilovanému programu.
- Strukturou velmi podobné technologii Silverlight (místo .NET je použita Java, místo XAML je použit JavaFX Script).



Obrázek 3.3.2.1 Struktura a návaznost JavaFX

Na obrázku 3.3.2.1 lze vidět, že JavaFx přímo komunikuje s běhovým prostředím a vytváří mezivrstvu pro komunikaci uživatelského rozhraní a programové logiky.

3.3.3 JavaFX Script

JavaFX Script [19] je statický typový, deklarativní skriptovací jazyk, jehož syntaxe je velmi podobná JavaScriptu. Jazyk je určen hlavně na tvorbu komplexního vzhledu uživatelských rozhraní.

- Kód JavaFX Scriptu je úspornější, než kód v Javě.
- Aplikace JavaFX je rychlejší, než klasický Java Applet.
- Plná podpora Swingových komponent Javy (Swing je knihovna pro vytváření grafického uživatelského vzhledu v Javě).
- Podpora 2D grafiky a přehrávání videa.

Příklad:

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
Stage {
    width: 250
    height: 80
    scene: Scene {
        content: Text {
            x: 10, y: 30
            content: "Hello World"
        }
    }
}
```

Takto vytvořený a zkompilovaný kód je následně zabalen do jar archívu a umístěn do HTML stránky, kde funguje podobně jako applet:

```
<html>
    <head>
    </head>
    <body>
        <script>
            javafx(
            {
                archive: "HelloApplet.jar",
                width: 150,
                height: 100,
                code: "HelloApplet",
            }
        );
        </script>
    </body>
</html>
```

3.4 Adobe Flash

Adobe Flash [20] je multiplatformní grafický program používaný pro vývoj interaktivních animací, reklam, prezentací a her. Flash dokáže manipulovat jak rastrovou, tak hlavně vektorovou grafikou, což z něj činí velmi mocný nástroj pro jednoduchou tvorbu grafických aplikací. Díky uložení vektorů má navíc výsledný program velmi malou velikost. Takto vytvořený program se přidá do těla

HTML stránky, je stažen na klienta a následně spuštěn ve webovém prohlížeči. Ke spuštění programu je nutné mít nainstalováno běhové prostředí – Adobe Flash Player.

Samotné aplikace Flash jsou vytvářeny pomocí objektově orientovaného jazyka ActionScript, který umožňuje vytvářet velké aplikace.

Právě malá velikost a rychlost aplikace je jedním z důvodů, proč je Flash jedním z nejvíce rozšířených RIA aplikací na světě.

Integrování do HTML stránky se provádí podobně jako u jiných aplikací přidaných do těla HTML stránky:

```
<object data="movie.swf" type="application/x-shockwave-flash>
    <param name="movie" value="movie.swf" />
</object>
```

Adobe Flash umožňuje:

- Vytváření grafické aplikace s pomocí vektorové grafiky.
- Vyváření grafických komponent a celkové „oživení“ a interaktivitu integrovanou přímo do webové stránky.
- Přehrávání audia, videa, a celkově velkou multimediální podporu.
- Velmi jednoduché vytváření designu a animací – programátor nepotřebuje znalost kódu.
- Používat Flash aplikace i mimo webový prohlížeč.
- Použití programovacího jazyka ActionScript umožňuje vytvoření programové logiky.
- Je platformově nezávislý.
- Podporuje 64-bitové systémy.

3.4.1 ActionScript

ActionScript [21] je objektově orientovaný programovací jazyk určený k vytváření programové logiky, která má být následně připojena ke grafickému vzhledu. ActionScript vychází ze standardizované verze JavaScriptu.

Samotné zpracování kódu je velmi jednoduché – vytvoření třídy:

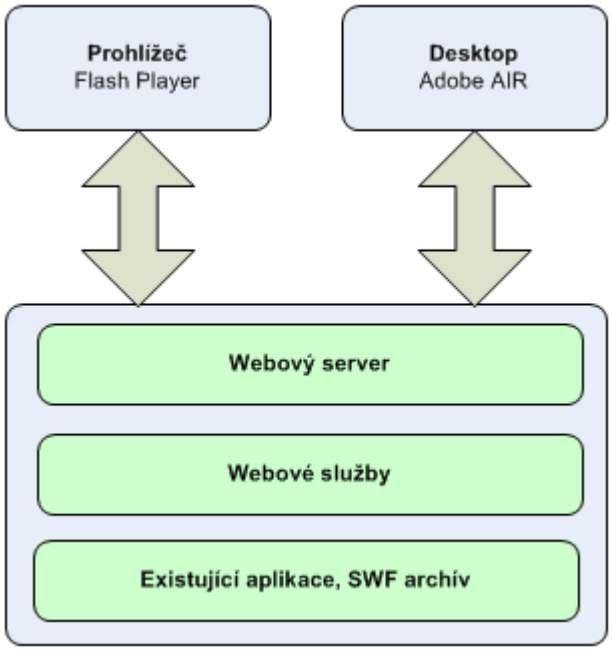
```
package ca.flashdev.hello
{
  public class HelloWorld
  {
    public function sayHello():String
    {
      var helloWorld:String = "Hello World!";
      return helloWorld;
    }
  }
}
```

A následné zavolání této třídy:

```
import ca.flashdev.hello.HelloWorld;
var classInstance:HelloWorld = new HelloWorld();
hello_txt.text = classInstance.sayHello();
```

3.5 Adobe Flex

Adobe Flex [22] je skupina technologií určená pro vývoj RIA. Zajímavé však je, že neobsahuje vlastní běhové prostředí. Pro běh Flex aplikací se proto využívají jiná běhová prostředí – pro prohlížeč Flash Player, pro desktop AIR.



Obrázek 3.5.1 Možnosti využití běhových prostředí

Na obrázku 3.5.1 je zobrazen celý proces. Nejprve je napsán kód, který je následně zkompilován do souboru swf. Zkompilovaný soubor je stažen ze serveru a je spuštěn buď v desktopovém běhovém prostředí AIR, nebo v prohlížeči v běhovém prostředí Flash Player.

Podle názvu by se dalo předpokládat jistou spojitost mezi Flash a Flex. Flex jako takový má ale s Adobe Flash společné pouze běhové prostředí, jinak se od flashe velmi liší. Aplikace je postupně skládána z jednotlivých komponent, celý program je tvořen objektově orientovaným jazykem. Zdrojové kódy jsou uloženy v textovém souboru na rozdíl od flashových binárních souborů.

Samotná aplikace je vyvíjena ve značkovacím jazyce MXML. Flex aplikace se skládá z různých komponent, které jsou reprezentovány třídami, tedy jsou objektově orientované, což umožňuje vytvářet i velké programy.

Díky jednotlivým komponentům vytvořeným v MXML se poměrně jednoduše vytváří také uživatelská rozhraní, která se dají lehce nadefinovat. (mají vlastnosti a parametry).

3.5.1 MXML

MXML [23] je objektově orientovaný jazyk, který obsahuje předvytvořené komponenty určené k vytváření grafických uživatelských rozhraní. Je založen na XML a zdrojový kód je vždy platný XML dokument.

Z následující ukázky kódu lze vidět, že názvy komponent jsou opravdu velmi intuitivní.

Takto vytvořený kód je při kompilaci převeden na ActionScript – jednotlivé komponenty v MXML obsahují ekvivalenty v ActionScriptu. A tento ActionScript je poté následně zkompilován do swf souboru.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
            layout="absolute" backgroundGradientColors="[#000011, #333333]">
    <mx:Label text="Hello World!" verticalCenter="0" horizontalCenter="0"
    fontSize="48" letterSpacing="1">
       <mx:filters>
          <mx:GlowFilter color="#ffffdd"/>
       </mx:filters>
    </mx:Label>
</mx:Application>
```

3.6 Adobe AIR (Adobe Integrated Runtime)

Adobe AIR (Adobe Integrated Runtime) [24] je univerzální běhové prostředí pro vytváření RIA aplikací, umožňující jejich fungování vně webového prohlížeče v desktopovém prostředí. Toto běhové prostředí umožňuje vytvářet aplikace napsané ve dvou variantách spojení kódů:

- HTML / JavaScript / CSS / AJAX
- Flash / Flex / ActionScript

Původní název běhového prostředí byl „Apollo“ a prostředí bylo určeno k podpoře Flexu. Následně byl projekt přejmenován na AIR.

Vzhledem k tomu, že AIR pracuje přímo na desktopu, získává několik výhod oproti RIA aplikacím běžících ve webovém prohlížeči. Tou nejzajímavější výhodou je přístup k lokálním souborům a souborovému systému. AIR díky tomuto přístupu umožňuje data ukládat přímo na disk na rozdíl od RIA aplikací, které většinu dat ukládají přímo na server. Díky tomu existuje možnost pracovat s aplikací i v offline režimu. AIR tedy může pracovat s lokální databází.

Kromě lokální databáze samozřejmě může AIR přistupovat k datům uloženým na serveru (přes webové služby) nebo k datům uloženým v XML souborech.

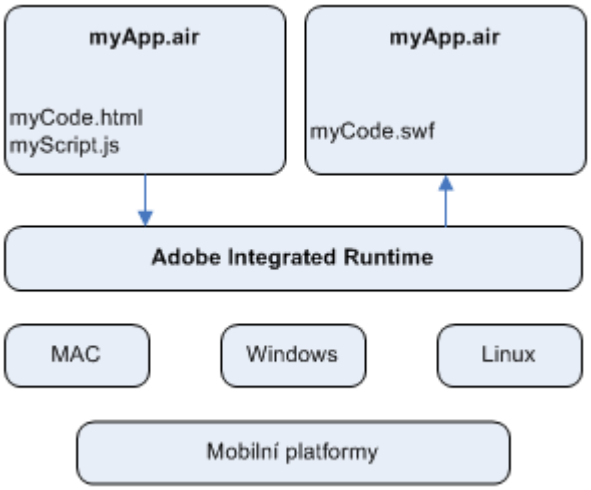
AIR se snaží přiblížit se ovládáním desktopovým aplikacím, jejich přirozené funkčnosti. Proto jednotlivé prvky a rysy jsou naprosto shodné s normálními desktopovými aplikacemi (tlačítka, menu).

Jak již bylo řečeno, AIR umožňuje používání JavaScriptu. Nabízí se možnost využití některých JavaScriptových frameworků. AIR ovšem běží v sandboxu kvůli většímu zabezpečení. Z tohoto důvodu funkce, které vyžadují dynamické generování kódu (jako například eval()), jsou v AIR zakázány. Právě proto byly některé frameworky upraveny tak, aby mohly s AIR spolupracovat.

K tomu, aby AIR na klientovi fungovalo, je potřeba mít nainstalován malý balíček běhového prostředí tak, jako když spouštím jiné RIA aplikace v prohlížeči, v němž je potřeba mít příslušný plugin.

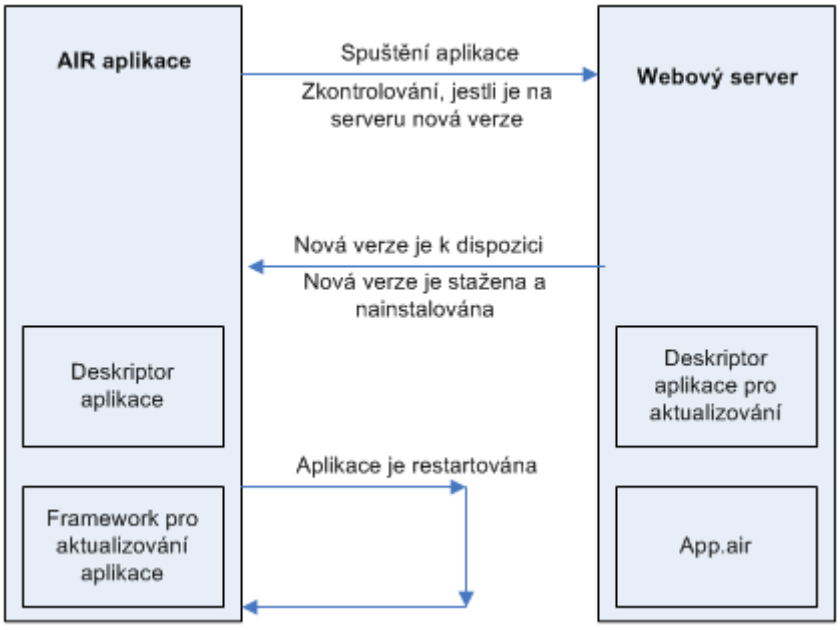
Oproti jiným běhovým prostředím je AIR velmi dobře uzpůsoben i pro fungování na mobilních platformách, což může být pro mnohé jiné RIA technologie problém. V AIR jsou však pro změnu platformy potřeba jen minimální úpravy.[25]

Na obrázku 3.6.1 je zobrazeno propojení rozhraní AIR s kódem. Nezáleží na tom, jestli je kód aplikace vyvíjen jedním nebo druhým způsobem, výsledek „myApp.air“ je použitelný na různých platformách.



Obrázek 3.6.1 Nezávislost platforem na AIR

Kromě řady možností nabízí AIR také zajímavou službu pro možnost automatické aktualizace daného desktopového programu.



Obrázek 3.6.2 Průběh aktualizace aplikace AIR

Na obrázku 3.6.2 je znázorněn průběh aktualizace. Aplikace zkontroluje předem určený server, a pokud je nalezena nová verze, je stažena a nainstalována. Poté je aplikace restartována.

Samotný kód aplikace se skládá ze dvou částí:

- Kód aplikace – zde je umístěna všech aplikační logika programu.

- Popisovač (Descriptor) – XML soubor ve kterém je nadefinováno vše, co se netýká přímo aplikační logiky. Jsou zde umístěny parametry pro velikost okna, parametry pro instalaci aplikace, spouštěný soubor, a vůbec parametry týkající se celkové struktury aplikace.

Hlavní rysy AIR jsou tedy:

- Podpora více platforem, a to i mobilních zařízení – stačí mít pouze nainstalováno AIR framework.
- Běh aplikace v sandboxu – zabezpečení klienta.
- Fungování vně webového prohlížeče – v desktopovém prostředí.
- Možnost přístupu k souborovému systému.
- Možnost využití lokální databáze na klientovi.
- Vývoj aplikace ve dvou možných jazykových kombinacích.
- Možnost automatických aktualizací.

3.6.1 SQLite

SQLite [26] je relativně malá databáze určená k používání hlavně u RIA aplikací. Požívá se k ukládání dočasných dat potřebných pro danou aplikaci a k jejímu celkové zkomfortnění . Právě její velikost z ní dělá velmi populární variantu pro tyto RIA aplikace. Využívá standardní SQL dotazovací jazyk, avšak některé jeho funkce jsou ořezané.

3.7 Porovnání technologií

Jednotlivé klady a zápory jsou přehledně zobrazeny a shrnuty v následujících tabulkách.

	AJAX	Silverlight	JavaFX	Flash	Flex
Windows	ANO	ANO	ANO	ANO	ANO
Mac OS X	ANO	ANO	ANO	ANO	ANO
Linux	ANO	NE	ANO	ANO	ANO
Podpora mobilních zařízení	ANO	WM	ANO	ANO	ANO
Build Format	.js	.xap	.fx	.swf	.swf/.air
Desktop/Prohlížeč	P	D/P	D/P	D/P	D/P
Ochrana zdrojového kódu	NE	ANO	ANO	NE	NE
Podpora 2D/3D	2D/3D	2D/3D	2D/3D	2D/3D	2D/3D

	AIR (HTML/AJAX)	AIR (Flash/ Flex)
Windows	ANO	ANO
Mac OS X	ANO	ANO
Linux	ANO	ANO
Podpora mobilních zařízení	ANO	ANO
Desktop/Prohlížeč	D	D
Ochrana zdrojového kódu	NE	NE
Podpora 2D/3D	2D/3D	2D/3D

4 Návrh AIR aplikace

Cílem je vytvořit aplikaci, která demonstruje rozsah AIR technologie. Není úkolem vytvořit rozsáhlý program, kde by se pravděpodobně většina funkcí opakovala. Taktéž cílem není porovnávání ostatních technologií použitých, nebo využívajících aplikaci. Naopak, smyslem aplikace je poukázat na výhody, přednosti a možnosti využití této technologie.

Pro tuto demonstraci bude vytvořen jednoduchý FTP klient s názvem FTPFileManager.

4.1 Specifikace

Aplikace FTPFileManager představuje jednoduchý program pro připojení, stahování, či nahrávání dat ze serveru nebo na něj. Zároveň udržuje aktivní účet aktuálně přihlášeného uživatele, který má možnost si ukládat nastavení daného připojení pro budoucí znovupoužití. Mezi hlavní funkce tedy patří možnost stahovat/nahrávat data z/na server, ovšem při využití minimální doby připojení k serveru.

4.2 Použité technologie

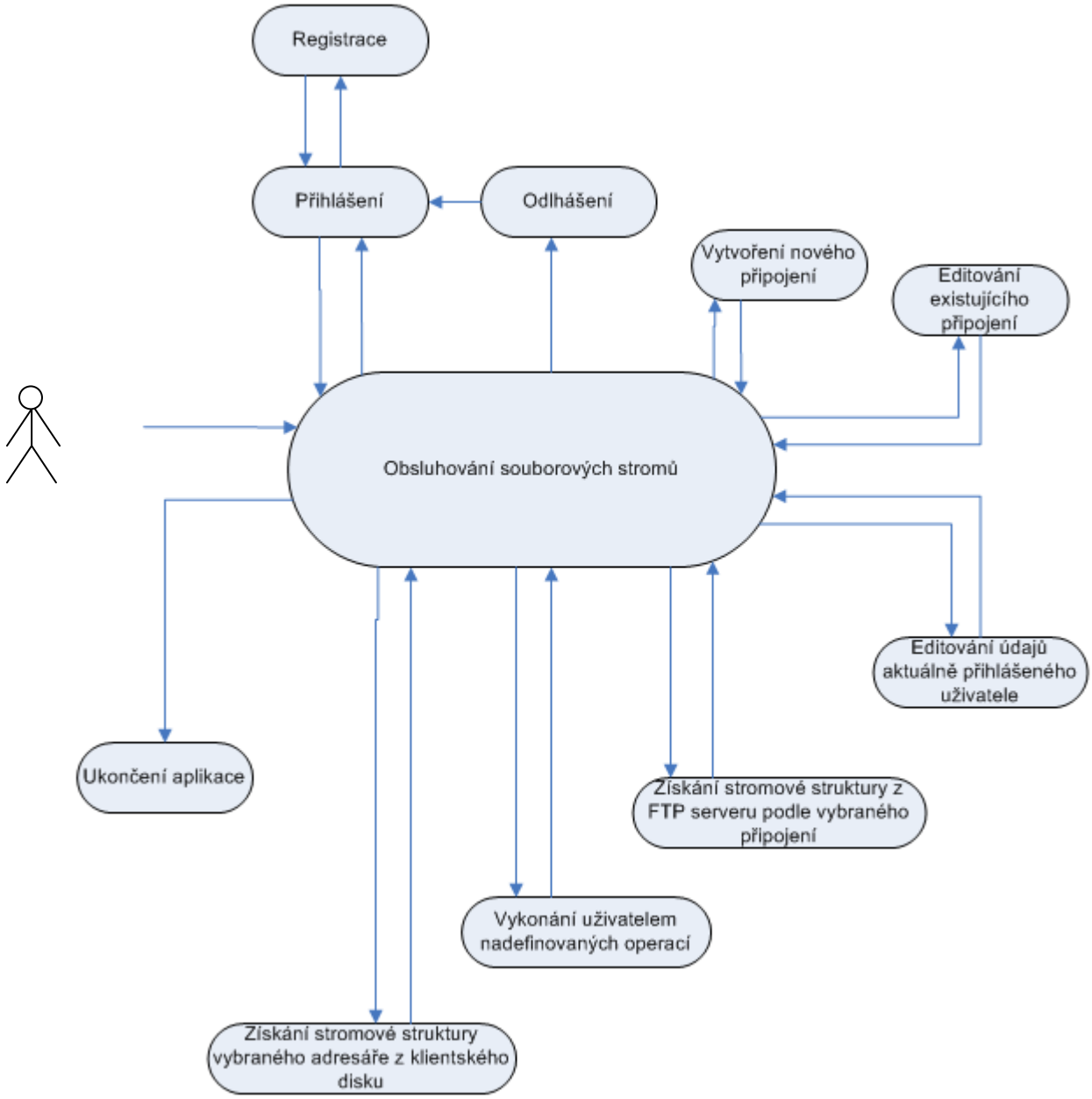
- Aplikace je vyvíjena ve spojení HTML / JavaScript / CSS / AJAX technologií a následně zkompilována do běhového prostředí AIR.
- Pro grafické rozhraní je použit Javascriptový framework ExtJS 2.2.
- Jako vývojové prostředí je použito Aptana Studio 1.5 [27], které umožňuje nainstalovat plugin pro testování či kompilování AIR aplikací.
- Jako běhové prostředí je využit AIR 1.5

4.3 Diagram případů užití

Na obrázku 4.3.1 máme diagram případů užití. Aplikace tedy funguje následovně:

- Uživatel spustí program.
- Pokud není uživatel aktuálně přihlášen, zobrazí se přihlašovací menu. (Program neodhlašuje po uzavření – uživatel sám musí zvolit akci odhlášení).
- Uživatel se přihlásí. Pokud nemá založen učet, zvolí Registrování nového účtu.
- Zobrazí se menu pro registraci nového účtu. Uživatel vyplní údaje a zaregistruje nový účet. Následně je vrácen na přihlašovací menu, kde se přihlásí pod vytvořeným účtem.
- Nyní má uživatel několik možností:
- Může otevřít v menu akci „Change user detail“ a upravit informace o svém účtu.
- Muže otevřit v menu akci „New connection“ a vyplnit údaje a vytvořit nové připojení.
- Může otevřít v menu akci a „Connection detail“, vybrat již existující připojení a to editovat.
- Dále může tlačítkem „Client Directory“ nastavit pracovní klientský adresář, jehož obsah se zobrazí v levém stromu „Client“.
- Pokud je vybráno připojení, může přes tlačítko „Server Directory“ získat adresářovou strukturu FTP serveru.
- Nyní může provádět operace nad danými stromy. Může, kopírovat, mazat, přesouvat a vytvářet adresáře – Tento krok nevyžaduje připojení k serveru. Provedené operace jsou zaznamenány a uloženy do databáze.
- Při stisku tlačítka „Execute“ se provedou operace, které uživatel nastavil mezi jednotlivými stromy.

- Nyní se může uživatel odhlásit, nebo ukončit aplikaci. Pokud aplikaci ukončí, zůstane při příštím spuštění přihlášen.



Obrázek 4.3.1 Diagram případů užití.

5 Implementace

V této kapitole se budeme zabývat samotnou implementací aplikace FTPFileManager. Bude zde zobrazen způsob řešení jakým daná aplikace funguje.

5.1 Vytvoření AIR aplikace

Pro samotné fungování AIR aplikací potřebujeme mít nainstalované běhové prostředí, které je ke stažení na oficiálních stránkách Adobe. Samotná instalace je velmi jednoduchá – jednoduchý průvodce nás provede celou instalací.

Následně potřebujeme vývojový nástroj. Taktéž na stránkách Adobe je zdarma ke stažení AIR SDK (Software Development Kit). K jeho instalaci je potřeba mít nainstalovánu Javu (jednoduše třeba Java virtual machine). AIR SDK umožní z příkazového řádku kompilovat vytvořený program.

Nyní vytvoříme adresář s názvem projektu, který bude obsahovat všechny použité soubory a knihovny.

Dále potřebujeme vytvořit XML soubor – deskriptor dané aplikace. V něm jsou uložena data o dané aplikaci. Rozměry, startovní soubor, verze aplikace, atd. Jednoduchý deskriptor může vypadat takto:

```
<?xml version="1.0" encoding="UTF-8"?>
        <application xmlns="http://ns.adobe.com/air/application/1.5">
        <id>examples.html.HelloWorld</id>
        <version>0.1</version>
        <filename>HelloWorld</filename>
        <initialWindow>
                <content>HelloWorld.html</content>
                <visible>true</visible>
                <width>400</width>
                <height>200</height>
        </initialWindow>
</application>
```

Následně zkopírujeme všechny potřebné „knihovny“, které bude naše aplikace využívat. Nejedná se o knihovny v pravém slova smyslu. Jsou to javascriptové soubory obsahující kód nebo předvytvořené komponenty pro zjednodušení vyvíjení.

Dále vytvoříme HTML stránku. Jedná se o klasickou HTML stránku bez jakékoli změny. Pouze bude mít v hlavičce odkaz na knihovny, které mají být obsaženy. Díky jednoduchému vývoji se stává běhové prostředí velmi zajímavé právě pro vývojáře webových stránek. Na dané stránce totiž můžou být využity CSS styly, Javascript a samozřejmě AJAX.

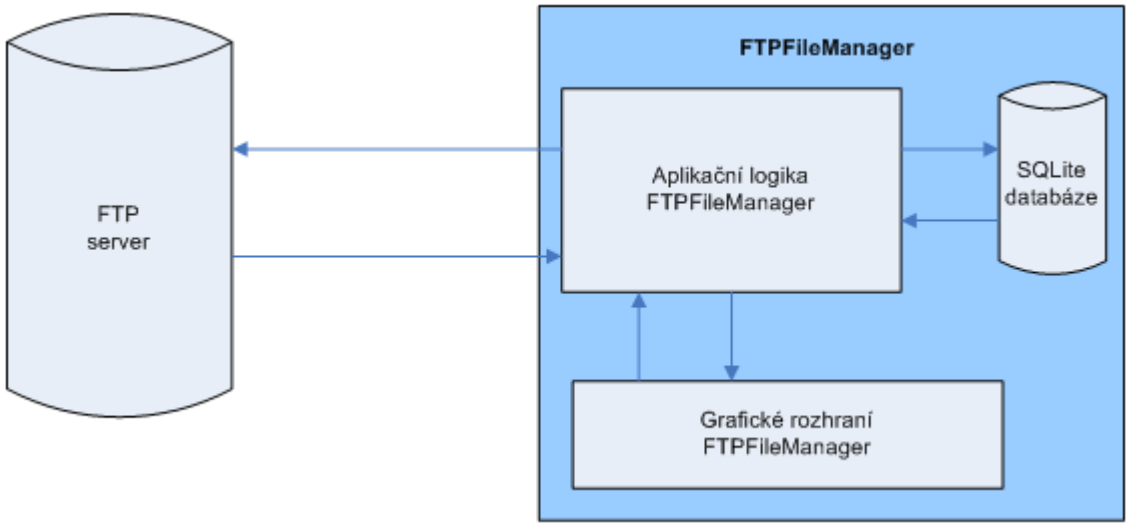
Nyní zbývá vytvoření instalačního souboru. Využijeme příkazů AIR SDK z příkazové řádky (adt -package) pro vytvoření tohoto souboru. K jeho vytvoření je však potřeba podpisový certifikát pro digitální podepsání dané aplikace.

Vytvořený instalační soubor je archív všech určených souborů a může být kdykoli rozbalen. Z toho plyne nechráněnost zdrojového kódu.

Tento způsob vývoje by ovšem byl velmi náročný hlavně na odlaďování vytvořeného kódu, proto využijeme nějaké vývojové prostředí. V našem případě (vývoj HTML/AJAX aplikace) máme k dispozici Adobe Dreamweaver s rozšířením Adobe AIR. Toto vývojové prostředí je ovšem placené,

proto využijeme již zmíněné Aptana Studio s pluginem Adobe AIR. Nabízí dostatečné možnosti, co se obsluhy projektu týče, a hlavně umožňuje odladit vytvořený Javascriptový kód.

5.2 Struktura aplikace FTPFileManager



Obrázek 5.1.1 Struktura aplikace FTPFileManager

Na obrázku 5.1.1 můžeme vidět strukturu aplikace FTPFileManager a její komunikaci jednotlivých komponent. Uživatel ovládá aplikaci přes grafické rozhraní, které komunikuje s aplikační logikou. K aplikaci je připojena SQLite databáze určená k uložení uživatelských dat a zaznamenání provedených operací. Nejsou zde uloženy žádné mnohatisícové záznamy, ostatně k tomu ani SQLite databáze není určena. Aplikační logika následně komunikuje přímo s FTP serverem.

Ze struktury aplikace je vidět, že je velmi podobná normálním 3-vrstvým desktopovým aplikacím – je odděleno grafické rozhraní, aplikační logika a úložiště dat.

5.3 Vývoj aplikace FTPFileManager

Kompletní dokumentaci k AIR lze nalézt na internetových stránkách [28]. AIR má poměrně dobré funkce pro ovládání lokálních souborů (tvorba, kopírování, mazaní atd). K souborům je přistupováno stejně jako k adresářům.

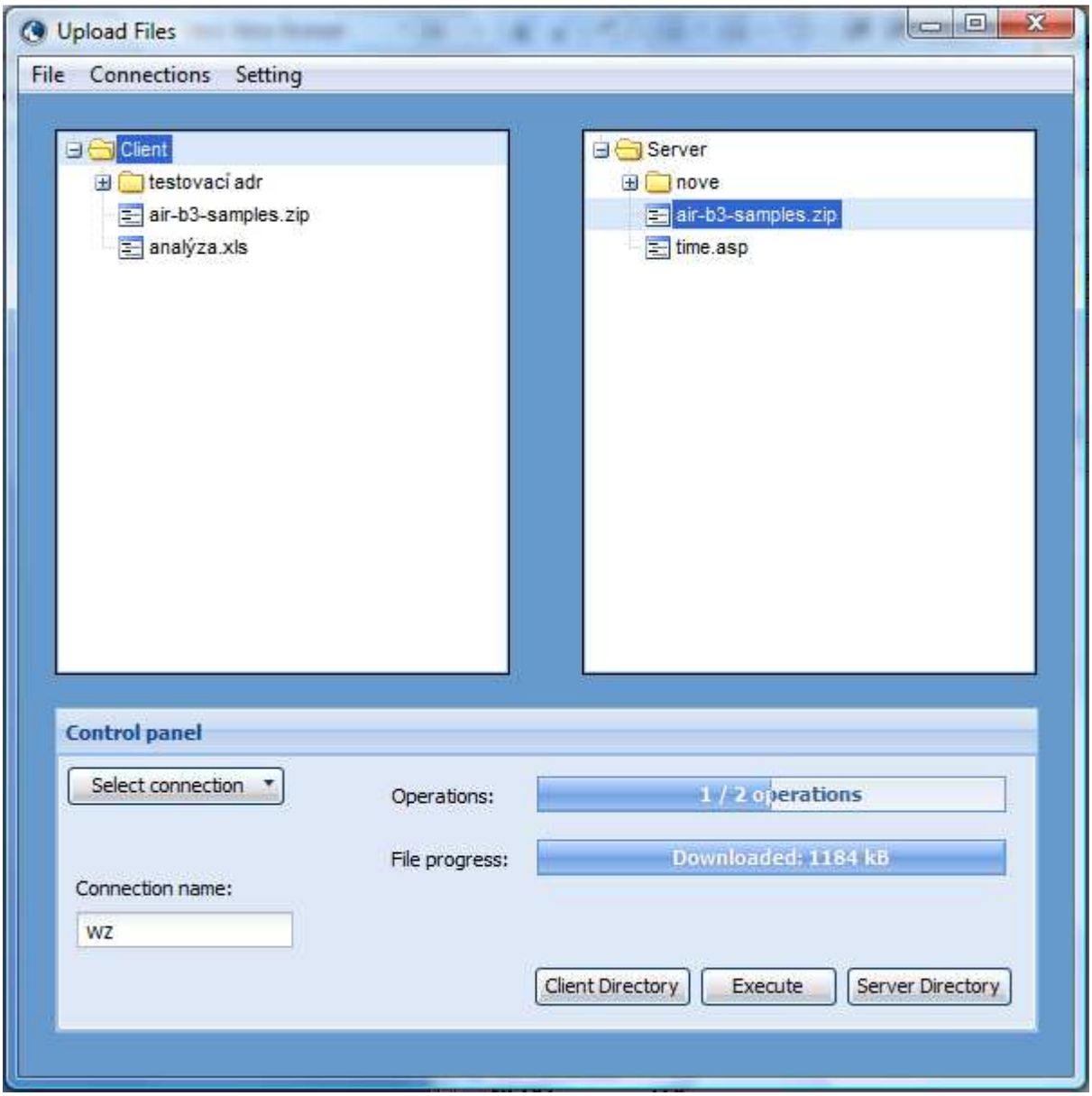
Taktéž k databázi je přistupováno obvyklým způsobem. Je vytvořeno spojení s databází (air.SQLConnection) a jsou volány dotazy v jazyce SQL do databáze.

Pro samotné vytvoření oken aplikace je použita třída air.NativeWindow.

Jediný problém tedy vzniká při komunikaci s FTP serverem. Adobe aktuálně nemá žádnou veřejně známou knihovnu pro komunikaci s FTP serverem, tudíž si musíme vytvořit vlastní obsluhu komunikace.

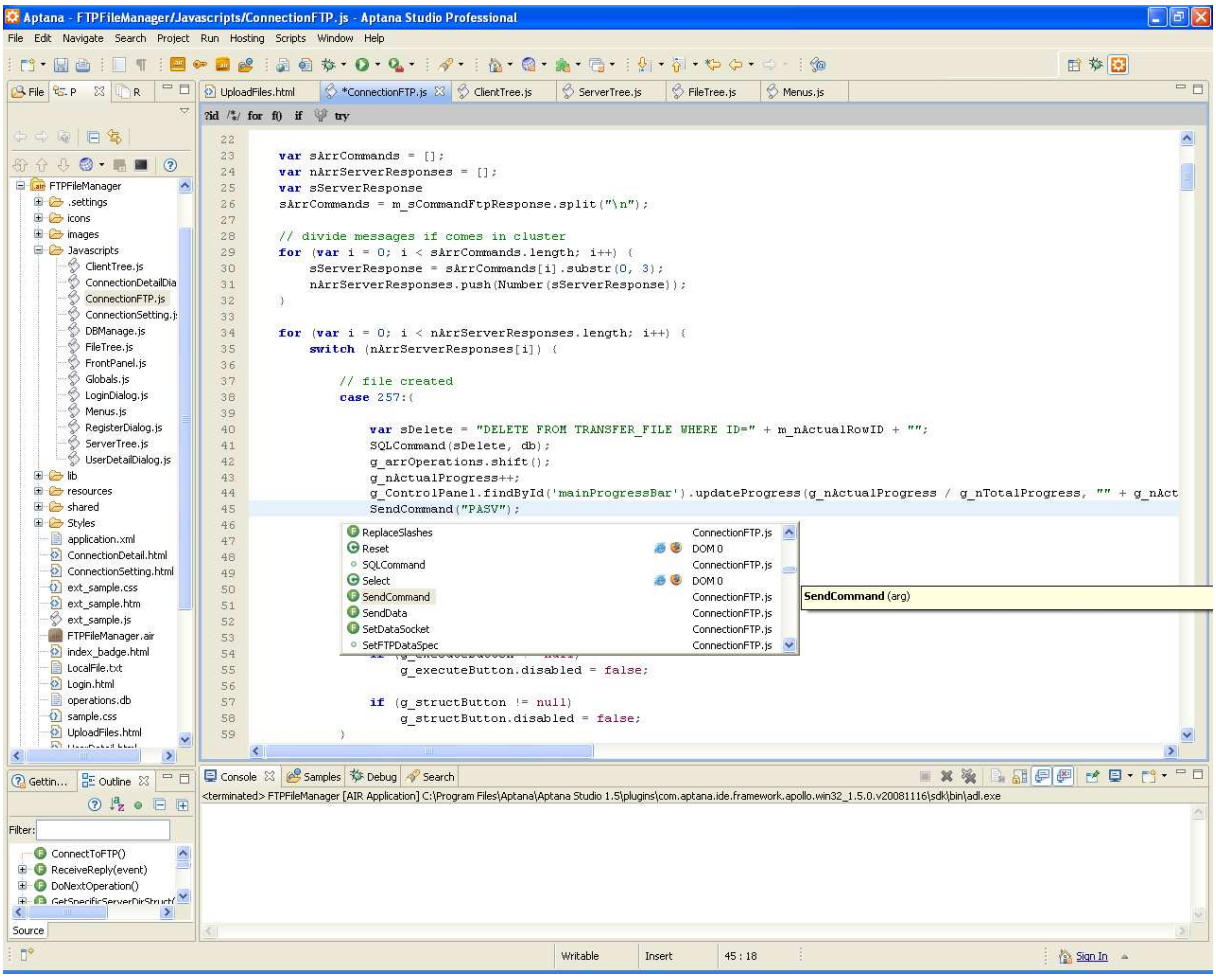
AIR nabízí využití socketů (air.Socket). FTP server komunikuje přes dva sockety (řídící a datový), proto na začátku komunikace otevřeme řídící socket a posíláme přes něj textové příkazy FTP serveru. FTP server vrací textové odpovědi na tyto příkazy. Každá odpověď začíná třímístným číslem. Podle toho můžeme poznat, jakou nám FTP server vrací odpověď, a zavoláme příslušné funkce, případně otevřeme datový socket, pokud se jedná o přenos dat.

Co se grafického rozhraní týče, je využit Javascriptový Framework ExtJS, který poskytuje komponenty vzhledově velmi podobné normálním desktopovým aplikacím, což ocení zejména uživatel. Náhled můžeme vidět na obrázku 5.3.1.



Obrázek 5.3.1 Grafické uživatelské rozhraní aplikace FTPFileManager

Co se vývojového prostředí týče, Aptana Studio poskytuje všechno potřebné pro jednoduchou obsluhu projektu tak, aby uživatel nemusel využívat příkazovou řádku, nebo podobných věcí. Díky podpoře AIR je vestavěna celkem zdařilá instellisence (nápovědu). Aptana dále nabízí možnost krokování v kódu pro odladění, což se zejména pro JavaScript hodí. V neposlední řádě je zde několik knihoven určených k odlaďování kódu jako třeba AIRIntrospector.js (podobný nástroji FireBug). Na obrázku 5.3.2 můžeme vidět náhled vývojového prostředí.



Obrázek 5.3.2 Vývojové prostředí Aptana Studio

5.4 Výhody použití AIR pro aplikaci FTPFileManager

- Implementace: Zkušený webový vývojář nepotřebuje znát k vývoji AIR aplikace jiné programovací jazyky, než které používá(HTML , Javascript), což se nedá říct o ostatních zde představených technologiích (JavaFX – využívá Javu, Silverlight – využívá .NET).
- Vzhled a chování aplikace je velmi podobný normálním desktopovým aplikacím. Hodně funkcí v AIR je určeno právě k vytvoření dojmu chování desktopové aplikace. Navíc framework ExtJS poskytuje velmi účinné a pěkné komponenty pro interakci a oživení aplikace.
- Možnost pracovat offline. Je to další náznak podoby s desktopovými aplikacemi. Aplikace FTPFileManager potřebuje připojení k serveru pouze ve chvíli, kdy stahuje souborovou strukturu FTP serveru, a potom ve chvíli, kdy provádí vykonání zadaných příkazů a přesunů vztahujících se k serveru. Oproti jiným FTP klientům tedy nepotřebuje udržovat spojení se serverem, i když si uživatel pouze prohlíží data obsažená na serveru. Tohle je velmi důležitý prvek, neboť ukazuje na zajímavou věc. Stále více začínají být rozšířené mobilní platformy, které taktéž nabývají na výkonu. Vzhledem k jejich mobilnosti ovšem samozřejmě není zaručeno permanentní připojení k Internetu. Proto možnost využití minimálního času pro

připojení je velmi důležitá. V AIR se dají navíc vyvinout aplikace, které dokonce vůbec nepotřebují připojení k internetu. Svým způsobem toto dané aplikaci odebírá status RIA, a stává se z ní plnohodnotná desktopová aplikace – nijak nekomunikuje s webem.

- Využití na mobilní platformě. AIR je možno využívat na velkém množství platforem s prakticky nezměněným zdrojovým kódem. Co se týče aplikace FTPFileManager, není optimalizovaná k použití na mobilní platformě. Je odzkoušena pouze na PC. Domnívám se, že na mobilní platformě by pravděpodobně nefungovala z důvodu grafického rozhraní (nastavení velikosti, a těžko odhadnout jak by se chovaly ExtJS komponenty). Ovšem aplikační logika, ovládání databáze a komunikace s FTP serverem by měly fungovat naprosto stejně. Právě platformová nezávislost dělá z AIR velmi silný a mocný nástroj.

5.5 Nevýhody použití AIR pro aplikaci FTPFileManager

- Ačkoliv AIR poskytuje množství funkcí a dokáže vytvořit velmi velký dojem desktopové aplikace, je to technologie, která se stále vyvíjí, proto ani AIR není dokonalý. AIR například nepodporuje více vláken. Taktéž sandbox dokáže velmi omezit. Dá se ovšem předpokládat, že spousta problémů bude v budoucnu odstraněna.
- Aplikace FTPFileManager neudržuje permanentní komunikaci s FTP serverem. Toto bylo jmenováno ve výhodách, ovšem je možnost, že zatímco si uživatel nastavuje operace, někdo jiný se přihlásí na FTP server a strukturu souboru úplně přehází. Uživatel se o tom dozví až ve chvíli, kdy bude chtít provést zaznamenané operace. Tohle je ovšem problém každého FTP klienta, zde se to ale projeví více právě z důvodu neaktualizace serverové struktury. Tohle není ani tak nevýhoda technologie AIR, jako spíše implementace samotné aplikace FTPFileManager. Takových a podobných chyb by se na této aplikaci dalo najít jistě hodně. Ovšem aplikace FTPFileManager má sloužit hlavně k demonstrativním účelům. Jistěže by se do ní dalo přidat také daleko větší množství funkčnosti, takto by se ale dala tato aplikace rozšiřovat dál a dál klidně několik let.

6 Závěr

Internetové technologie se objevují všude tam, kde je Internet. Ovšem například při cestování je dostupnost připojení k Internetu stále problém. Stejně tak mobilní stanice nemají neustálou možnost připojení. Druhý důvod vzniku RIA aplikací je komfort uživatele, kdy je využíváno znalostí, které může potencionální uživatel mít pro obsluhu daného programu. Tyto znalosti uživatel nabral využíváním desktopových aplikací. Výsledkem těchto dvou důvodů je vytvoření hybridního typu webové aplikace fungujícího mimo webový prohlížeč.

Adobe AIR není zase tak známá technologie jako je například Silverlight. Cílem práce bylo seznámit čtenáře s touto technologií, ukázat přednosti i negativa, zobrazit rozsah, který tato technologie dokáže pokrýt a následně vytvořit srovnání vůči dalším podobným technologiím.

Možné rozšíření práce může spočívat v porovnání většího počtu technologií, případně v pokusech o vytvoření podobných aplikací pro dané technologie.

Neméně důležitá je otázka budoucího vývoje. S velkým vzestupem a vývojem mobilních stanic, s jehož pokračováním můžeme téměř jistě počítat i do budoucna, že bude pokračovat i do budoucna, se dá předpokládat, že nezávislost a přenositelnost zdrojového kódu na platformě bude velmi důležitý prvek v boji o mobilní platformy.

Literatura

[1] : WWW stránky. Fat-client - Wikipedia, otevřená encyklopedie, http://en.wikipedia.org/wiki/Fat_client Naposledy navštíveno 24.4.2010

[2]: WWW stránky. Thin-client – Wikipedia, otevřená encyklopedie, http://en.wikipedia.org/wiki/Thin_client Naposledy navštíveno 24.4.2010

[3]: WWW stránky. Webová aplikace – Wikipedia, otevřená encyklopedi, http://cs.wikipedia.org/wiki/Webov%C3%A1_aplikace Naposledy navštíveno 24.4.2010

[4]: WWW stránky. Rfc 2616 – HyperText Tranfer Protocol – HTTP http://tools.ietf.org/html/rfc2616 Naposledy navštíveno 24.4.2010

[5]: WWW stránky. Rfc 959 – File Transfer Protocol – FTP, http://tools.ietf.org/html/rfc959 Naposledy navštíveno 24.4.2010

[6]: WWW stránky. Stránky JavaScriptu, http://www.javascript.com/ Naposledy navštíveno 24.4.2010

[7]: WWW stránky. Popis DOM, http://www.w3.org/DOM/ Naposledy navštíveno 24.4.2010

[8]: WWW stránky. Obrázek zjednodušené struktury DOM, http://net.tutsplus.com/tutorials/javascript-ajax/javascript-and-the-dom-series-lesson-1/ Naposledy navštíveno 24.4.2010

[9]: WWW stránky. Stránky JSON, http://www.json.org/ Naposledy navštíveno 24.4.2010

[10]: WWW stránky. Oficální stránky ExtJS http://www.extjs.com/ Naposledy navštíveno 24.4.2010

[11]: WWW stránky. Oficiální stránky AJAX, http://www.ajax.org/ Naposledy navštíveno 24.4.2010

[12]: WWW stránky. Obrázek znázornění komunikace dat mezi serverem a klientem, http://www.eioba.com/a984/ajax_a_new_approach_to_web_applications Naposledy navštíveno 24.4.2010

[13]: WWW stránky. Oficiální stránky technologie Silverlight http://www.silverlight.net/ Naposledy navštíveno 24.4.2010

[14]: WWW stránky. Detailní popis architektury WPF http://msdn.microsoft.com/en-us/library/bb404713%28VS.95%29.aspx Naposledy navštíveno 24.4.2010

[15]: WWW stránky. Oficiální stránky technologie Moonlight http://www.mono-project.com/Moonlight Naposledy navštíveno 24.4.2010

[16]:WWW stránky. Oficiální stránky Java
http://java.sun.com/ Naposledy navštíveno 24.4.2010

[17]: WWW stránky. Oficiální stránky Java Applet
http://java.sun.com/applets/ Naposledy navštíveno 24.4.2010

[18]: WWW stránky. Oficiální stránky JavaFX
http://javafx.com/ Naposledy navštíveno 24.4.2010

[19]: WWW stránky. Oficiální stránky JavaFX Script
http://www.sun.com/software/javafx/script/ Naposledy navštíveno 24.4.2010

[20]: WWW stránky. Oficiální stránky Adobe
http://www.adobe.com/ Naposledy navštíveno 24.4.2010

[21]: WWW stránky. Stránky zabývající se vývojem a popisem ActionScript
http://www.actionscript.org/ Naposledy navštíveno 24.4.2010

[22]: WWW stránky. Oficiální stránky Adobe Flex
http://flex.org/ Naposledy navštíveno 24.4.2010

[23]: WWW stránky. MXML specifikace
http://opensource.adobe.com/wiki/display/flexsdk/MXML+2009 Naposledy navštíveno 24.4.2010

[24]: WWW stránky. Oficiální stránky firmy Adobe
http://www.adobe.com/ Naposledy navštíveno 24.4.2010

[25]: WWW stránky. Demonstrační video zobrazující chování AIR na různých platformách
http://www.youtube.com/watch?v=22vicDlzmkI Naposledy navštíveno 22.4.2010

[26]: WWW stránky. Oficiální stránky SQLite
http://sqlite.org/ Naposledy navštíveno 22.4.2010

[27]: WWW stránky. Oficiální stránky Aptana
http://www.aptana.org/ Naposledy navštíveno 30.4.2010

[28]: WWW stránky. Dokumentace k AIR API
http://help.adobe.com/en_US/air/reference/html/ Naposledy navštíveno 30.4.2010

Seznam příloh

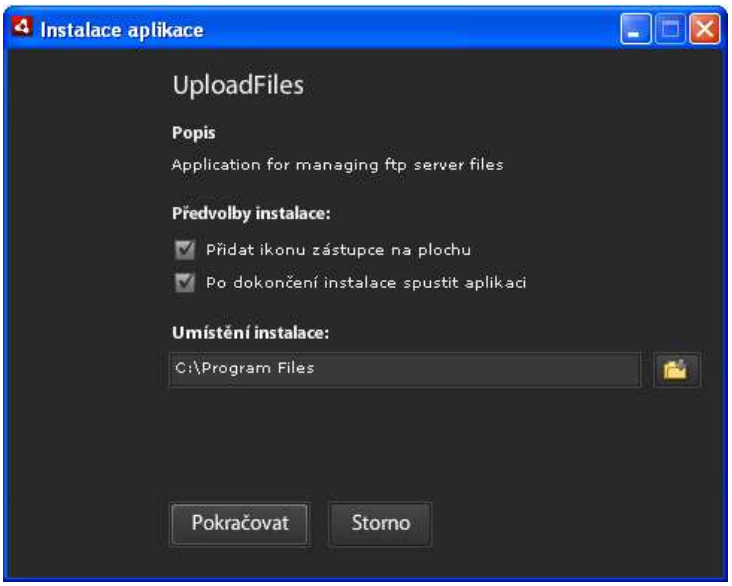
Dodatek A Popis zprovoznění ukázkové aplikace.
Dodatek B CD se zdrojovými kódy.

Dodatek A

Popis zprovoznění ukázkové aplikace

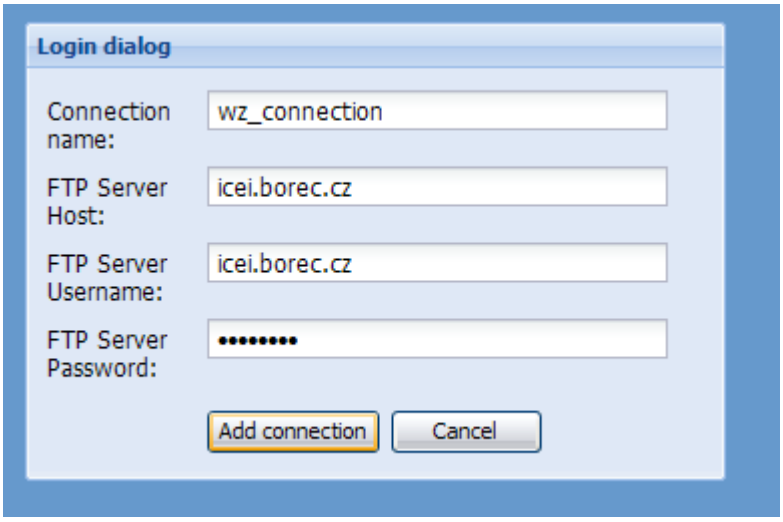
Nainstalujeme běhové prostředí Adobe AIR
http://get.adobe.com/air/

Spustíme soubor FTPFileManager.air a postupujeme podle nabízených kroků.

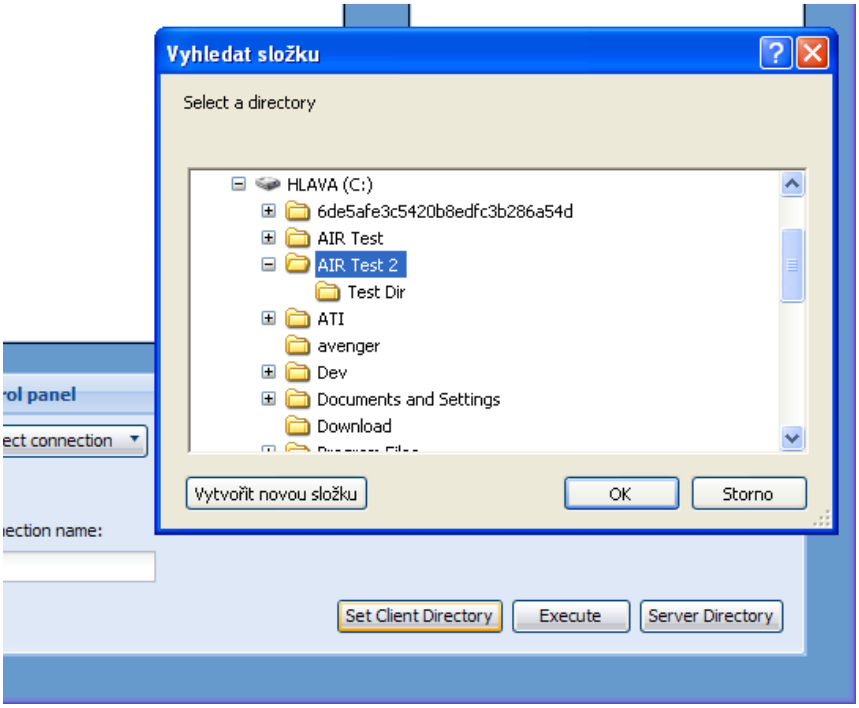


Zprovoznění aplikace:
Po přihlášení vytvoříme nové připojení:

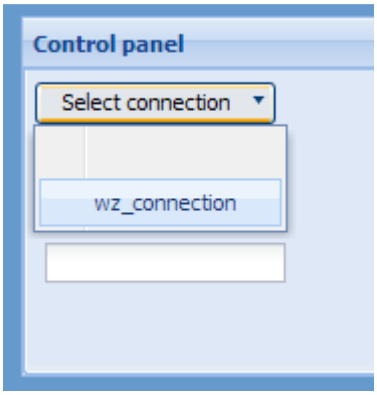
A vyplníme potřebná data pro připojení.



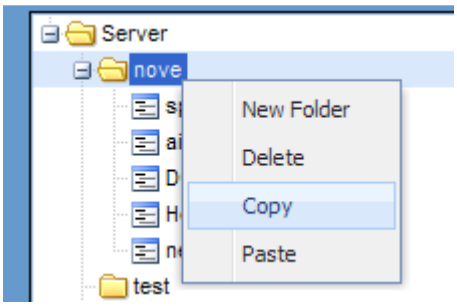
Nastavíme klientský adresář:



Vybereme vytvořené připojení z nabídky připojení:



Přes talčítko”Server Directory” získáme adresářovou strukturu z FTP serveru. Nyní můžeme se soubory manipulovat:



Ve chvíli, kdy jsme plně spokojení s provedenými operacemi, zmačkneme tlačítko “Execute” a všechny zaznamenané operace se provedou.

Celkový vzhled poté vypadá takto.